과제 3#. LR Parser

소프트웨어학부 20180325 임성현

1. 과제 내용

- LR 문법과 LR Parsing Table을 이용해 LR Parser 프로그램을 작성하였다.

2. 문제 해결 방법

- 강의교안 1-29~1-31, 1-35~1-36 슬라이드를 참고하였다.

- int value[1000] 를 별도로 만들어 구현하였다.

- yylex() 함수 내 while (ch == ' ' || ch == '\t' || ch == '\n') 를 while (ch == ' ' || ch == '\t') // || ch == '\n') 로 수정하여 입력받도록 하였다. 또한, yytext[i] = 0; 로 되어있어 value[] 에 값을 읽어들이지 못하였다. 따라서, 해당 코드를 주석처리하였다.

- 기존 코드에서 발생한 warning 메시지를 제거했다.

- LLDB (디버거)를 활용하였다.

3. 실행 결과

- 테스트 케이스를 통과하는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- 테스트 케이스를 통과하지 못하는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

4. 결론

- 과제 명세에 맞게, *부호가 없는 여러자릿수의 정수, 소괄호, +, \* 등을 포함하는 여러가지 수식이 잘 처리되는지 실험* 하였다.

- LR Parser의 작동 원리를 알게 되었다.

5. 문제점

- ‘3(100+20)’ 같은 수식은 계산할 수 없다.

- ‘3+3 3’ 와 같이 수식 사이에 빈 칸이 있는 경우 계산할 수 없다.

- float형 수식은 계산할 수 없다.

6. 원시 프로그램 (소스코드)

#include <ctype.h> //isdigit()

#include <stdio.h>

#include <stdlib.h> // exit()

int yyparse();

int yylex();

void shift();

void reduce();

void yyerror();

void lex\_error();

#define NUMBER 256

#define STAR 258

#define RPAREN 260

#define EXPRESSION 0

#define FACTOR 2

#define PLUS 257

#define LPAREN 259

#define END 261

#define TERM 1

#define ACC 999

int action[12][6] = {

{5, 0, 0, 4, 0, 0}, {0, 6, 0, 0, 0, ACC}, {0, -2, 7, 0, -2, -2},

{0, -4, -4, 0, -4, -4}, {5, 0, 0, 4, 0, 0}, {0, -6, -6, 0, -6, -6},

{5, 0, 0, 4, 0, 0}, {5, 0, 0, 4, 0, 0}, {0, 6, 0, 0, 11, 0},

{0, -1, 7, 0, -1, -1}, {0, -3, -3, 0, -3, -3}, {0, -5, -5, 0, -5, -5}};

int go\_to[12][3] = {{1, 2, 3}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},

{8, 2, 3}, {0, 0, 0}, {0, 9, 3}, {0, 0, 10},

{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}};

int prod\_left[7] = {0, EXPRESSION, EXPRESSION, TERM, TERM, FACTOR, FACTOR};

int prod\_length[7] = {0, 3, 1, 3, 1, 3, 1};

int stack[1000];

int top = -1;

int sym;

// 1-35 슬라이드

char yytext[32];

int yylval;

int value[1000];

// 기존 코드에서 컴파일러 warning 으로 인해 void main()에서 int main()으로 변경

int main() {

printf("입력하는 수식 : ");

yyparse();

}

int yyparse() {

int i;

stack[++top] = 0; // initial state

sym = yylex();

do {

i = action[stack[top]][sym - 256]; // get relation

if (i == ACC)

// printf("success !\n");

printf("= %d\n", value[1]);

else if (i > 0)

shift(i); // shift

else if (i < 0)

reduce(-i); // reduce

else

yyerror();

} while (i != ACC);

return 0; // 기존 코드에서 컴파일러 warning 으로 인해 추가

}

void push(int i) { stack[++top] = i; }

void shift(int i) {

push(i);

value[top] = yylval;

sym = yylex();

}

void reduce(int i) {

int old\_top;

top -= prod\_length[i];

old\_top = top;

push(go\_to[stack[old\_top]][prod\_left[i]]);

switch (i) {

case 1:

value[top] = value[old\_top + 1] + value[old\_top + 3];

break;

case 2:

value[top] = value[old\_top + 1];

break;

case 3:

value[top] = value[old\_top + 1] \* value[old\_top + 3];

break;

case 4:

value[top] = value[old\_top + 1];

break;

case 5:

value[top] = value[old\_top + 2];

break;

case 6:

value[top] = value[old\_top + 1];

break;

default:

yyerror("parsing table error");

break;

}

}

void yyerror() {

printf("syntax error\n");

exit(1);

}

// get\_token()

int yylex() {

static char ch = ' ';

int i = 0;

// 기존 코드 수정 : 모든 수식을 읽은 후 값을 처리하지 못해서 주석 처리

// i.e. 1-34 슬라이드에서 s7 -> s5로 넘어가는 순간

while (ch == ' ' || ch == '\t') // || ch == '\n')

ch = getchar();

if (isdigit(ch)) {

yytext[i] = ch;

ch = getchar();

yylval = atoi(yytext);

while (isdigit(ch)) {

yytext[i] = ch;

ch = getchar();

yylval = yylval \* 10 + atoi(yytext);

};

// 기존 코드 수정 : value[]에 값을 읽지 못하므로 주석 처리

// yytext[i] = 0;

// 기존 코드 수정 : 여러자리수 계산시 atoi() 위치 조정

// yylval = atoi(yytext);

return (NUMBER);

} else if (ch == '+') {

ch = getchar();

return (PLUS);

} else if (ch == '\*') {

ch = getchar();

return (STAR);

} else if (ch == '(') {

ch = getchar();

return (LPAREN);

} else if (ch == ')') {

ch = getchar();

return (RPAREN);

// 기존 코드 수정 : '\n' 입력시 더 이상 입력받지 않고 종료

} else if (ch == EOF || ch == '\n')

return (END);

else

lex\_error();

return 0; // 기존 코드에서 컴파일러 warning 으로 인해 추가

}

void lex\_error() {

printf("syntax error: invalid character\n");

exit(1);

}